Chapter 6:   Installing Products Using UPD

This chapter guides you through installing products from a **UPS/UPD** product distribution node using the **UPD** command **`upd install`**. **UPD** is the most efficient and widely-used product installation method on machines running **UPS/UPD**. The installation parameters are set in the local node's **UPD** configuration. The aspects of the **UPD** configuration that you as a product installer need to be aware of are described in section 4.3 *What You Need to Know before Modifying Your UPD Configuration*; the configuration file itself is described in detail in Chapter 32: *The UPD Configuration File*.

6.1 Quick Installation Instructions

If the following statements are true for you, or at least if you think they are, then go to section 1.1 *Install a Product from KITS; UPS/UPD Exists on Target Machine*.

- You don't know any **UPS** details, and furthermore, you don't *want* to!
- Your machine is configured properly for **UPS/UPD**, and the defaults are set appropriately.
- You want to install a product from KITS.
- A product tar file made for your OS exists in KITS, and is set as "current" so that you don't need to know the version or flavor, you just need the product name.
- The product is configured in KITS such that it can be installed in a very straightforward manner.

There are examples in section 6.5 *Examples*. If you need more information, read on!

6.2 The upd install Command

The **`upd install`** command performs the following functions:

- retrieves the specified product instance, and by default its dependencies, from a distribution node
- installs the product, and by default its dependencies, on the user node according to the local **UPD** configuration
- unwinds the product(s) if transferred in tar format
- declares the product, and by default its dependencies, to the database specified in the local **UPD** configuration. You may pass options to this local database declaration.
- either resolves dependencies (if the **`-X`** option is specified) or prints to screen the commands you will need to issue in order to do so

6.2.1 Command Syntax

The full description of the **`upd install`** command and the options it takes can be found in the reference section 24.8 *upd install*. The command syntax, showing some commonly used options, is:

```
% upd install [<chainFlag>] [-G "<options>"]  [-h <host>] \
  [-H <flavor>] [-q <qualifiers>] [-X] [-z <databaseList>] \
  [<other options>] <product> [version]
```

6.2.2 Passing Options to the Local ups declare Command

The **`-G`** option allows you to pass **UPS** options to the local **`ups declare`** command, which gets called internally by **`upd install`**. **`-G`** supports multiple options. The elements valid for use with **`-G`** include **`<product>`**, **`<version>`** and the following subset of the **`ups declare`** options:

`-A <nodeList>`, **`-c`**, **`-d`**, **`-D <origin>`**, **`-f <flavor>`**, **`-g <chainName>`**, **`-n`**, **`-o`**, **`-O "<flagList>"`**, **`-p "<description>"`**, **`-q "<[+|?]qualifier1:[+|?]qualifier2 ...>"`**, **`-t`**, **`-z <databaseList>`**, **`-0`**, **`-1`**, **`-2`**, **`-3`**, **`-4`**

This feature is most commonly used to declare a chain to the product, e.g., the "current" chain:

```
% upd install -G "-c" [<other options>] <product> [<version>]
```

The **`-G`** construction can also be used to reset identifying information like flavor and qualifiers. For example, to download the OSF1+V3 version of a product with qualifier **`oldxyz`**, but declare it locally as OSF1 with qualifier **`newxyz`**, use the **`-f`** (or **`-H`**), **`-q`** and **`-G`** options as shown:

```
% upd install -H OSF1+V3 -q oldxyz -G "-f OSF1 -q newxyz" \
  [<other options>] <product> [<version>]
```

6.3 How UPD Selects the Database

6.3.1 Database Selection Algorithm

`upd install` runs **`ups declare`** on the local node to declare the product instance to a local database. The local **UPD** configuration is always used to determine the database into which products must be installed/declared. If your **UPS** installation includes only one database, that is the one into which all declarations will go (assuming that the **UPD** configuration used by that database points to it, which is generally the case).

If there are multiple databases, **`upd install`** has to determine the database for the declaration. In a nutshell, it:

1) picks a starting database

   To pick the starting database, **UPD** first looks to see if the **`-z <databaseList>`** option is specified on the **`upd install`** command line. If so, **UPD** picks the first database listed there. If not, **UPD** picks the first database listed in $PRODUCTS.

2) finds the **UPD** configuration file to which the database points[1]

3) looks in this **UPD** configuration to see where to install and declare the product

If your local **UPS/UPD** installation is particularly complicated, it might be useful to verify that your $PRODUCTS variable includes all the databases used by the **UPD** configuration(s), and only those. If not, it is possible to declare products into a database not listed in $PRODUCTS, which generally is undesirable.

6.3.2 Database Selection for Dependencies

This works the same way as it does for the main product. For each dependency in turn, **UPD** looks in the **UPD** configuration file designated by the first database it encounters. From the **UPD** configuration, it determines the database in which to declare the dependent product, and where to install the product files. (If the database already contains a declaration for the same instance of the product, the product does not get reinstalled/redeclared.)

1. The location of the `updconfig` file is set via the keyword UPD_USERCODE_DIR in the database's `dbconfig` file.

6.3.3 Selecting a Database for Development or Testing

For development and/or testing purposes, it is often convenient to install products in your own products area and declare them in your own database, separate from the working database(s) on your system. Setting up your own database is discussed in section 12.8.2 *Adding a New Database and/or Products Area*.

If you're working in AFS space or in an NIS cluster with its own common NFS-mounted database, also see TN0091 *Configuring a Local UPS Database (While Still Using the Centrally Supported AFS database)* at `http://www.fnal.gov/docs/TN/tn0091.html`., or section 13.2 *Configuring a Local Database to Work With AFS*.

☞ Even if you've prepended your database path to $PRODUCTS, when you're ready to install a product and declare it in your database, remember that you may need to use the **`-z <database>`** option in the **`upd install`** command, as discussed above.

6.4 Checklist for Installing a Product using UPD

The procedural list below is a full checklist for a product installation, including pre- and post-install checks. The checks are not strictly necessary, the list simply provides guidelines for monitoring a product installation. Having said that, it's always a good idea to at least get a "snapshot" of your database before and after each product installation to aid in troubleshooting in the event the installation is not entirely successful. Include options/arguments as necessary in the suggested commands.

1) Run **`upd list`** to verify that the desired product instance is available on the server.

2) Run **`upd depend`** to list the product's dependencies (lists both required and optional by default).

3) Run **`upd depend -R`** to list only its required dependencies (compare to the **`upd depend`** output to determine the optional ones).

4) Run **`ups list`** to see which if any of the dependencies already exist in the local target database.

5) Run **`upd install`** to install the product instance and, as desired, its required and optional dependencies.

☞ When testing/troubleshooting, you might want to use the **`-i`** option to ignore errors, or the **`-v`**(**`vvv`**) option to produce verbose output.

6) Run **`ups declare -c`** if you want the parent product to be "current" but you did not include **`-G "-c"`** in **`upd install`**.

7) Run commands to resolve dependencies, if indicated in the output of the **`upd install`** command.

8) Run **`ups list`** for the parent product and dependencies to verify the declarations.

9) Setup the parent product and test that it works.

6.5 Examples

6.5.1 Install a Product Using Default Database

This illustrates the simplest case: installing the current instance of a product for the best-match flavor of the target machine, and letting **UPS** determine the target database using $PRODUCTS. For this example, we choose a product with no dependencies. First, check the default instance on the server:

```
% upd list -K+ teledata
```

```
   "teledata" "v1_0" "NULL" "" "current"
```

Check which instances already exist in the database(s) listed in $PRODUCTS:

```
% ups list -aK+ teledata
```

(*if no output, then no instances*)

Install the default instance. We are not passing any arguments to the local **`ups declare`** command (no **`-G`** option).

```
% upd install teledata
```

```
   informational: installed teledata v1_0.
   informational: product teledata has an INSTALL_NOTE;
                                          you     should     read
   /export/home/t1/aheavey/upsII/products/teledata/v1_0//up
   s/INSTALL_NOTE.
```

Read the `INSTALL_NOTE` file to see if you need to do anything (we'll not document this part since it is product-specific). Next, verify that the instance is now declared in $PRODUCTS:

```
% ups list -aK+ teledata
```

```
   "teledata" "v1_0" "NULL" "" ""
```

Redeclare the instance with the current chain:

```
% ups declare -c teledata v1_0
```

Verify that the instance is now declared as current:

```
% ups list -aK+ teledata
   "teledata" "v1_0" "NULL" "" "current"
```

6.5.2 Install a Product, Specifying Database

Perform the installation normally, (as shown in section 6.5.1 *Install a Product Using Default Database*) but include the `-z` option, e.g.,:

```
% upd install -z $MYDB teledata [<other options>]
```

or

```
% upd install -z /path/to/my/database teledata [<other options>]
```

assuming $MYDB is set to `/path/to/my/database`. The main product and all of its dependencies, if any, are installed and declared according to the **UPD** configuration to which the specified database points.

6.5.3 Install a Product and All Dependencies

By default, **`upd install`** installs the specified (parent) product and all of its dependencies. It checks for the presence of the dependencies as discussed in section 6.3 *How UPD Selects the Database*, and installs each as necessary, skipping over the ones already there. Make sure that you include neither the **`-j`** nor the **`-R`** option on the **`upd install`** command line. The use of these two options is illustrated in the following sections, 6.5.4 *Install a Product and No Dependencies* and 6.5.5 *Install a Product and Required Dependencies Only*.

To illustrate how **UPD** handles dependencies that are already installed versus those that aren't, we'll first install two of **pine**'s six dependencies separately, and then install **pine** v4_05 itself. We start with an empty database. First list **pine**'s dependencies:

```
% upd depend pine
   pine v4_05 -f IRIX+6 -z /ftp/upsdb -g current
   |__ispell v3_1b -f IRIX+6 -z /ftp/upsdb -g current
   |__ximagetools v4_0 -f NULL -z /ftp/upsdb -g current
      |__imagelibs v1_0 -f IRIX+6 -z /ftp/upsdb
      |__imagemagick v4_04 -f IRIX+6 -z /ftp/upsdb
      |__xfig v3_20 -f IRIX+6 -z /ftp/upsdb<---- already there
      |__xanim v2_70_64 -f IRIX+6 -z /ftp/upsdb
```

We'll install **xfig** and **xanim** ahead of time, without specifying a chain:

```
% upd install xfig -H IRIX+6
```

```
% upd install xanim -H IRIX+6
```

Install **pine** and chain it to current:

```
% upd install pine -H IRIX+6 -G "-c"
```

```
   informational: xanim v2_70_64 already exists on local node,
   skipping.
   informational: xfig v3_20 already exists on local node,
   skipping.
   informational: installed imagemagick v4_04.
   informational: installed imagelibs v1_0.
   informational: installed ximagetools v4_0.
   informational: installed ispell v3_1b.
   informational: installed pine v4_05.
```

Take a snapshot of the post-installation database:

% ups list -aK+

```
"imagelibs" "v1_0" "IRIX+6" "" "current"
"imagemagick" "v4_04" "IRIX+6" "" "current"
"ispell" "v3_1b" "IRIX+6" "" "current"
"pine" "v4_05" "IRIX+6" "" "current"
"xanim" "v2_70_64" "IRIX+6" "" ""
"xfig" "v3_20" "IRIX+6" "" ""
"ximagetools" "v4_0" "NULL" "" "current"
```

Notice that all the products are chained to current except the two that were preinstalled with no chain. The installer should go ahead and declare them current, too, so that the main product recognizes them as dependencies[1].

A second example illustrates how **UPS/UPD** resolves chains and discusses the effect of the **`-X`**, **`-s`** and **`-v`** options. We install **www** version v2_7b. First list its dependencies:

% upd depend www v2_7b

```
www v2_7b -f SunOS+5 -z /ftp/upsdb
|__ximagetools v4_0 -f NULL -z /ftp/upsdb -g current
|  |__imagelibs v1_0 -f SunOS+5 -z /ftp/upsdb
|  |__imagemagick v4_04 -f SunOS+5 -z /ftp/upsdb
|  |__xfig v3_20 -f SunOS+5 -z /ftp/upsdb
|  |__xanim v2_70_64 -f SunOS+5 -z /ftp/upsdb
|__xpdf v0_5 -f SunOS+5 -z /ftp/upsdb -g current
```

Let’s assume none of these products exists in our local database, and run the install:

% upd install www v2_7b

```
informational: installed xpdf v0_5.
informational: installed xanim v2_70_64.
informational: installed xfig v3_20.
informational: installed imagemagick v4_04.
informational: installed imagelibs v1_0.
informational: installed ximagetools v4_0.
informational: installed www v2_7b.
informational: product www has an INSTALL_NOTE;    you should
read

/local/ups/prd/upd/mengel/test/out_products/www/SunOS-5/v2_7
b/ups/INSTALL_NOTE.
Execute the following to resolve chains:
ups declare -f SunOS+5 -q "" -g current xpdf v0_5
        -z /local/ups/prd/upd/mengel/test/out_ups_database
```

1. Typically dependencies are defined by chain rather than by version.

```
   ups declare -f NULL -q "" -g current ximagetools v4_0
           -z /local/ups/prd/upd/mengel/test/out_ups_database
```

If we had included the **`-X`** option, **UPS/UPD** would have executed these two **`ups declare`** commands. If we had run it with the **`-s`** option, which just lists what the command would do, the output would have looked like:

```
   informational: would have installed xpdf v0_5.
   informational: would have installed xanim v2_70_64.
   informational: would have installed xfig v3_20.
   informational: would have installed imagemagick v4_04.
   informational: would have installed imagelibs v1_0.
   informational: would have installed ximagetools v4_0.
   informational: would have installed www v2_7b.
   upd install would have succeeded.
```

If we had run it with the **`-v`** (verbose) option and **`-X`**, we would have seen (output edited for brevity):

```
informational: beginning install of xpdf.
informational:                                   transferred
/ftp/products/xpdf/v0_5/SunOS+5/xpdf_v0_5_SunOS+5
        from fnkits.fnal.gov to

/local/ups/prd/upd/mengel/test/out_products/xpdf/SunOS-5/v0_
5
informational:                                   transferred
/ftp/products/xpdf/v0_5/SunOS+5/xpdf_v0_5_SunOS+5/ups
/.
        from fnkits.fnal.gov to

/local/ups/prd/upd/mengel/test/out_products/xpdf/SunOS-5/v0_
5/ups
informational:                                   transferred
/ftp/products/xpdf/v0_5/SunOS+5/xpdf_v0_5_SunOS+5.tab
le
        from fnkits.fnal.gov:/ to

/local/ups/prd/upd/mengel/test/out_ups_database/xpdf/v0_5.ta
ble.new
informational: ups declare succeeded
...(plus similar output for each remaining product)...
```

6.5.4 Install a Product and No Dependencies

Perform the installation normally (as shown in section 6.5.1 *Install a Product Using Default Database*), but include the **`-j`** option, e.g.,:

```
% upd install -j <product> [<version>] [<other options>]
```

The specified product gets installed, but none of its dependencies do.

6.5.5 Install a Product and Required Dependencies Only

In this example we'll install the product **exmh** version v2_0_2, flavor IRIX+6 (the default for our system) and its required dependencies only. We perform the installation normally, but include the **`-R`** option to specify "required dependencies only". First take a snapshot of the local database into which the product and its required dependencies will be declared:

```
% ups list -aK+
```

```
"imagelibs" "v1_0" "IRIX+6" "" "current"
"imagemagick" "v4_04" "IRIX+6" "" "current"
"ispell" "v3_1b" "IRIX+6" "" "current"
```

```
   "pine" "v4_05" "IRIX+6" "" "current"
   "xanim" "v2_70_64" "IRIX+6" "" ""
   "xfig" "v3_20" "IRIX+6" "" ""
   "ximagetools" "v4_0" "NULL" "" "current"
```

Check the full dependency list for this product:

```
% upd depend exmh
```

```
   exmh v2_0_2 -f IRIX+6 -z /ftp/upsdb -g current
   |__expect v5_25 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |  |__tk v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |__tk v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |  |__tcl v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |__tcl v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |__mh v6_8_3c -f IRIX+6 -z /ftp/upsdb -g current
   |  |__mailtools v2_3 -f NULL -z /ftp/upsdb -g current
   |__mimetools v2_7a -f IRIX+6 -z /ftp/upsdb -g current
   |__glimpse v3_0a -f IRIX+6 -z /ftp/upsdb -g current
   |__www v3_0 -f NULL -z /ftp/upsdb -g current
   |  |__lynx v2_8_1 -f IRIX+6 -z /ftp/upsdb -g current
   |__ispell v3_1b -f IRIX+6 -z /ftp/upsdb -g current
```

Of these dependencies, only **ispell** v3_1b is already installed locally. List the required dependencies:

```
% upd depend -R exmh
```

```
   exmh v2_0_2 -f IRIX+6 -z /ftp/upsdb -g current
   |__expect v5_25 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |  |__tk v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |__tk v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |  |__tcl v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |__tcl v8_0_2 -f IRIX+5 -z /ftp/upsdb -j  -g current
   |__mh v6_8_3c -f IRIX+6 -z /ftp/upsdb -g current
   |  |__mailtools v2_3 -f NULL -z /ftp/upsdb -g current
   |__mimetools v2_7a -f IRIX+6 -z /ftp/upsdb -g current
```

Glimpse, **www**, **lynx** and **ispell** are not in this list and therefore **`upd install -R`** should not install them. Install **exmh** (output not shown):

```
% upd install exmh v2_0_2 -H IRIX+6 -R
```

Now take a post-installation snapshot of the local database:

```
% ups list -aK+
```

```
   "exmh" "v2_0_2" "IRIX+6" "" "current"
   "expect" "v5_25" "IRIX+5" "" "current"
   "imagelibs" "v1_0" "IRIX+6" "" "current"
   "imagemagick" "v4_04" "IRIX+6" "" "current"
   "ispell" "v3_1b" "IRIX+6" "" "current"
   "mailtools" "v2_3" "NULL" "" "current"
   "mh" "v6_8_3c" "IRIX+6" "" "current"
   "mimetools" "v2_7a" "IRIX+6" "" "current"
   "pine" "v4_05" "IRIX+6" "" "current"
   "tcl" "v8_0_2" "IRIX+5" "" "current"
   "tk" "v8_0_2" "IRIX+5" "" "current"
   "xanim" "v2_70_64" "IRIX+6" "" ""
   "xfig" "v3_20" "IRIX+6" "" ""
   "ximagetools" "v4_0" "NULL" "" "current"
```

Notice that **exmh**, all of its required dependencies, and none of its optional ones are listed (except **ispell** which was already there).